A Lanczos approach to the inverse square root of a large and sparse matrix

Artan Boriçi

Paul Scherrer Institute

CH-5232 Villigen PSI

Artan.Borici@psi.ch

Abstract

I construct a Lanczos process on a large and sparse matrix and use the results of this iteration to compute the inverse square root of the same matrix. The example used here comes from the theory of chiral fermions on the lattice.

1 Introduction

Quantum Chromodynamics (QCD) is a theory of strong interactions, where the chiral symmetry plays a mayor role. There are different starting points to formulate a lattice theory with exact chiral symmetry, but all of them must obey the Ginsparg-Wilson condition [1]:

$$\gamma_5 D^{-1} + D^{-1}\gamma_5 = a\gamma_5\alpha^{-1}, \tag{1}$$

where a is the lattice spacing, D is the lattice Dirac operator and α^{-1} is a local operator and trivial in the Dirac space.

A candidate is the overlap operator of Neuberger [2]:

$$D = 1 - A(A^\dagger A)^{-1/2}, \qquad A = M - aD_W \tag{2}$$

where M is a shift parameter in the range $(0, 2)$, which I have fixed at one and D_W is the Wilson-Dirac operator,

$$D_W = \frac{1}{2}\sum_{\mu}[\gamma_\mu(\partial^*_\mu + \partial_\mu) - a\partial^*_\mu\partial_\mu] \tag{3}$$

and ∂_μ and ∂^*_μ are the nearest-neighbor forward and backward difference operators, which are covariant, i.e. the shift operators pick up a unitary 3 by 3 matrix with determinant one. These small matrices are associated with the links of the lattice and are oriented positively. A set of such matrices forms a "configuration". $\gamma_\mu, \mu = 1, \ldots, 5$ are 4 by 4 matrices related to the spin of the particle. Therefore, if there are N lattice points, the matrix is of order $12N$. A restive symmetry of the matrix A that comes from the continuum is the so called $\gamma_5 - symmetry$ which is the Hermiticity of the $\gamma_5 A$ operator.

The computation of the inverse square root of a matrix is reviewed in [3]. In the context of lattice QCD there are several sparse matrix methods, which are developed recently. [4, 5, 6, 7, 8]. I will focus here on a Lanczos method similar to [7].

2 The Lanczos Algorithm

The Lanczos iteration is known to approximate the spectrum of the underlying matrix in an optimal way and, in particular, it can be used to solve linear systems [9].

Let $Q_n = [q_1, \ldots, q_n]$ be the set of orthonormal vectors, such that

$$A^\dagger A Q_n = Q_n T_n + \beta_n q_{n+1}(e^{(n)}_n)^T, \qquad q_1 = \rho_1 b, \qquad \rho_1 = 1/||b||_2 \tag{4}$$

where T_n is a tridiagonal and symmetric matrix, b is an arbitrary vector, and β_n a real and positive constant. $e^{(n)}_m$ denotes the unit vector with n elements in the direction m.

By writing down the above decomposition in terms of the vectors $q_i, i = 1, \ldots, n$ and the matrix elements of T_n, I arrive at a three term recurrence that allows to compute these

vectors in increasing order, starting from the vector q_1. This is the *LanczosAlgorithm*:

$$\begin{aligned}
&\beta_0 = 0,\ \rho_1 = 1/||b||_2,\ q_0 = o,\ q_1 = \rho_1 b \\
&for\ i = 1, \ldots \\
&\quad v = A^\dagger A q_i \\
&\quad \alpha_i = q_i^\dagger v \\
&\quad v := v - q_i\alpha_i - q_{i-1}\beta_{i-1} \\
&\quad \beta_i = ||v||_2 \\
&\quad if \beta_i < tol,\ n = i,\ end\ for \\
&\quad q_{i+1} = v/\beta_i
\end{aligned} \tag{5}$$

where *tol* is a tolerance which serves as a stopping condition.

The Lanczos Algorithm constructs a basis for the Krylov subspace [9]:

$$\text{span}\{b, A^\dagger A b, \ldots, (A^\dagger A)^{n-1} b\} \tag{6}$$

If the Algorithm stops after n steps, one says that the associated Krylov subspace is invariant.

In the floating point arithmetic, there is a danger that once the Lanczos Algorithm (polynomial) has approximated well some part of the spectrum, the iteration reproduces vectors which are rich in that direction [9]. As a consequence, the orthogonality of the Lanczos vectors is spoiled with an immediate impact on the history of the iteration: if the algorithm would stop after n steps in exact arithmetic, in the presence of round off errors the loss of orthogonality would keep the algorithm going on.

3 The Lanczos Algorithm for solving $A^\dagger A x = b$

Here I will use this algorithm to solve linear systems, where the loss of orthogonality will not play a role in the sense that I will use a different stopping condition.

I ask the solution in the form

$$x = Q_n y_n \tag{7}$$

By projecting the original system on to the Krylov subspace I get:

$$Q_n^\dagger A^\dagger A x = Q_n^\dagger b \tag{8}$$

By construction, I have

$$b = Q_n e_1^{(n)}/\rho_1, \tag{9}$$

Substituting $x = Q_n y_n$ and using (4), my task is now to solve the system

$$T_n y_n = e_1^{(n)}/\rho_1 \tag{10}$$

Therefore the solution is given by

$$x = Q_n T_n^{-1} e_1^{(n)}/\rho_1 \tag{11}$$

This way using the Lanczos iteration one reduces the size of the matrix to be inverted. Moreover, since T_n is tridiagonal, one can compute y_n by short recurences.

If I define:

$$r_i = b - A^\dagger A x_i, \quad q_i = \rho_i r_i, \quad \tilde{x}_i = \rho_i x_i \tag{12}$$

where $i = 1, \ldots$, it is easy to show that

$$\begin{aligned} &\rho_{i+1}\beta_i + \rho_i\alpha_i + \rho_{i-1}\beta_{i-1} = 0 \\ &q_i + \tilde{x}_{i+1}\beta_i + \tilde{x}_i\alpha_i + \tilde{x}_{i-1}\beta_{i-1} = 0 \end{aligned} \tag{13}$$

Therefore the solution can be updated recursively and I have the following *Algorithm1*

for solving the system $A^\dagger Ax = b$*:*

$$
\begin{aligned}
&\beta_0 = 0,\ \rho_1 = 1/||b||_2,\ q_0 = o,\ q_1 = \rho_1 b \\
&for\ i = 1, \ldots \\
&\quad v = A^\dagger A q_i \\
&\quad \alpha_i = q_i^\dagger v \\
&\quad v := v - q_i\alpha_i - q_{i-1}\beta_{i-1} \\
&\quad \beta_i = ||v||_2 \\
&\quad q_{i+1} = v/\beta_i \\
&\quad \tilde{x}_{i+1} = -\frac{q_i + \tilde{x}_i\alpha_i + \tilde{x}_{i-1}\beta_{i-1}}{\beta_i} \\
&\quad \rho_{i+1} = -\frac{\rho_i\alpha_i + \rho_{i-1}\beta_{i-1}}{\beta_i} \\
&\quad r_{i+1} := q_{i+1}/\rho_{i+1} \\
&\quad x_{i+1} := y_{i+1}/\rho_{i+1} \\
&\quad if \frac{1}{|\rho_{i+1}|} < tol,\ n = i,\ end\ for
\end{aligned}
\tag{14}
$$

4 The Lanczos Algorithm for solving $A^\dagger Ax = b$

Now I would like to compute $x = (A^\dagger A)^{-1/2} b$ and still use the Lanczos Algorithm. In order to do so I make the following observations:

Let $(A^\dagger A)^{-1/2}$ be expressed by a matrix-valued function, for example the integral formula [3]:

$$(A^\dagger A)^{-1/2} = \frac{2}{\pi}\int_0^\infty dt(t^2 + A^\dagger A)^{-1} \tag{15}$$

From the previous section, I use the Lanczos Algorithm to compute

$$(A^\dagger A)^{-1} b = Q_n T_n^{-1} e_1^{(n)}/\rho_1 \tag{16}$$

It is easy to show that the Lanczos Algorithm is shift-invariant. i.e. if the matrix $A^\dagger A$ is shifted by a constant say, t^2, the Lanczos vectors remain invariant. Moreover, the corresponding Lanczos matrix is shifted by the same amount.

This property allows one to solve the system $(t^2 + A^\dagger A)x = b$ by using the same Lanczos iteration as before. Since the matrix $(t^2 + A^\dagger A)$ is better conditioned than $A^\dagger A$,

it can be concluded that once the original system is solved, the shifted one is solved too. Therefore I have:

$$(t^2 + A^\dagger A)^{-1} b = Q_n (t^2 + T_n)^{-1} e_1^{(n)} / \rho_1 \tag{17}$$

Using the above integral formula and puting everything together, I get:

$$x = (A^\dagger A)^{-1/2} b = Q_n T_n^{-1/2} e_1^{(n)} / \rho_1 \tag{18}$$

There are some remarks to be made here:

a) As before, by applying the Lanczos iteration on $A^\dagger A$, the problem of computing $(A^\dagger A)^{-1/2} b$ reduces to the problem of computing $y_n = T_n^{-1/2} e_1^{(n)} / \rho_1$ which is typically a much smaller problem than the original one. But since $T_n^{1/2}$ is full, y_n cannot be computed by short recurences. It can be computed for example by using the full decomposition of T_n in its eigenvalues and eigenvectors; in fact this is the method I have employed too, for its compactness and the small overhead for moderate n.

b) The method is not optimal, as it would have been, if one would have applied it directly for the matrix $(A^\dagger A)^{1/2}$. By using $A^\dagger A$ the condition is squared, and one looses a factor of two compared to the theoretical case!

c) From the derivation above, it can be concluded that the system $(A^\dagger A)^{1/2} x = b$ is solved at the same time as the system $A^\dagger A x = b$.

d) To implement the result (18), I first construct the Lanczos matrix and then compute

$$y_n = T_n^{-1/2} e_1^{(n)} / \rho_1 \tag{19}$$

To compute $x = Q_n y_n$, I repeat the Lanczos iteration. I save the scalar products, though it is not necessary.

Therefore I have the following *Algorithm2 for solving the system* $(A^\dagger A)^{1/2}x = b$*:*

$$
\begin{aligned}
&\beta_0 = 0,\ \rho_1 = 1/||b||_2,\ q_0 = o,\ q_1 = \rho_1 b \\
&for\ i = 1, \dots \\
&\quad v = A^\dagger A q_i \\
&\quad \alpha_i = q_i^\dagger v \\
&\quad v := v - q_i\alpha_i - q_{i-1}\beta_{i-1} \\
&\quad \beta_i = ||v||_2 \\
&\quad q_{i+1} = v/\beta_i \\
&\quad \rho_{i+1} = -\frac{\rho_i\alpha_i + \rho_{i-1}\beta_{i-1}}{\beta_i} \\
&\quad if \frac{1}{|\rho_{i+1}|} < tol,\ n = i,\ end\ for \\
&Set\ (T_n)_{i,i} = \alpha_i,\ (T_n)_{i+1,i} = (T_n)_{i,i+1} = \beta_i, otherwise\ (T_n)_{i,j} = 0 \\
&y_n = T_n^{-1/2} e_1^{(n)}/\rho_1 = U_n \Lambda_n^{-1/2} U_n^T e_1^{(n)}/\rho_1
\end{aligned}
\tag{20}
$$

$$
\begin{aligned}
&q_0 = o,\ q_1 = \rho_1 b,\ x_0 = o \\
&for\ i = 1, \dots, n \\
&\quad x_i = x_{i-1} + q_i y_n^{(i)} \\
&\quad v = A^\dagger A q_i \\
&\quad v := v - q_i\alpha_i - q_{i-1}\beta_{i-1} \\
&\quad q_{i+1} = v/\beta_i
\end{aligned}
$$

where by o I denote a vector with zero entries and U_n, Λ_n the matrices of the egienvectors and eigenvalues of T_n. Note that there are only four large vectors necessary to store: q_{i-1}, q_i, v, x_i.

5 Testing the method

I propose a simple test: I solve the system $A^\dagger A x = b$ by applying twice the *Algorithm*2, i.e. I solve the linear systems

$$(A^\dagger A)^{1/2} z = b, \quad (A^\dagger A)^{1/2} x = z \tag{21}$$

in the above order. For each approximation x_i, I compute the residual vector

$$r_i = b - A^\dagger A x_i \tag{22}$$

The method is tested for a SU(3) configuration at $\beta = 6.0$ on a $8^3 16$ lattice, corresponding to an order 98304 complex matrix A.

In Fig.1 I show the norm of the residual vector decreasing monotonically. The stagnation of $||r_i||_2$ for small values of *tol* may come from the accumulation of round off error in the 64-bit precision arithmetic used here.

This example shows that the tolerance line is above the residual norm line, which confirms the expectation that *tol* is a good stopping condition of the *Algorithm*2.

6 Acknowledgement

The author would like to thank the organizers of this Workshop for the kind hospitality at Wuppertal.

References

[1] P. H. Ginsparg and K. G. Wilson, Phys. Rev. D 25 (1982) 2649.

[2] H. Neuberger, Phys. Lett. B 417 (1998) 141, Phys. Rev. D 57 (1998) 5417.

[3] N. J. Higham, Proceedings of "Pure and Applied Linear Algebra: The New Generation", Pensacola, March 1993.

[4] B. Bunk, Nucl.Phys.Proc.Suppl. B63 (1998) 952.

[5] H. Neuberger, Phys. Rev. Lett. 81 (1998) 4060.

[6] R. G. Edwards, U. M. Heller and R. Narayanan, Nucl.Phys. B540 (1999) 457-471.

[7] A. Boriçi, Phys.Lett. B453 (1999) 46-53.

[8] P. Hernandez, K. Jansen, and L. Lellouch, hep-lat/9909026

[9] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 1989. This is meant as a general reference with original references included therein.

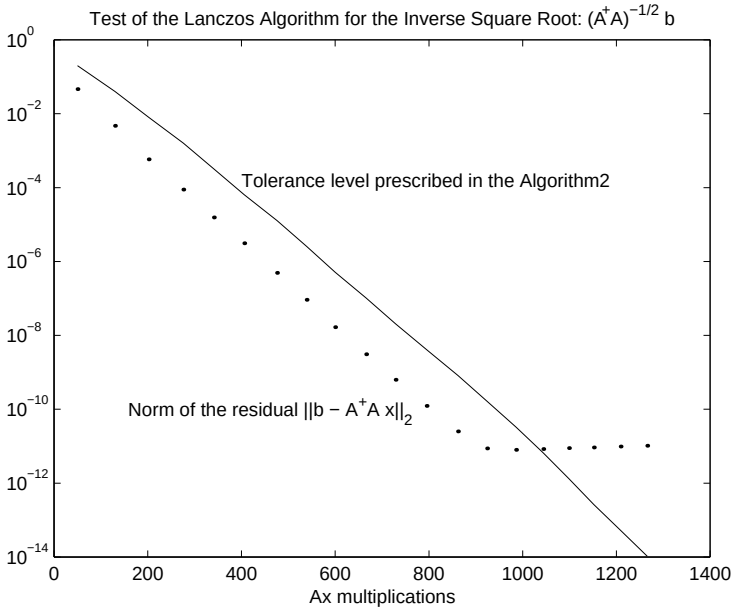

Figure 1: The dots show the norm of the residual vector, whereas the line shows the tolerance level set by *tol* in the *Algorithm*2.